第5章 DIV+CSS

DIV+CSS是Web标准中常用术语之一，是一种网页设计的布局方法，这种布局方法相对于代码层次混乱、样式与结构杂糅的表格定位方式而言，它带来了全新的布局理念，真正实现了内容与表现的分离。本章将帮助读者理解DIV元素定位和CSS布局理念，介绍CSS盒模型理论，掌握DIV+CSS常见网页布局的设计技巧，能利用CSS代码对网页文字、图像、背景、导航、超链接、表格及表单元素进行美化，从而设计出精美的网页作品。

本章学习要点：

- DIV元素定位
- 盒模型理论
- CSS布局理念
- DIV+CSS常见布局
- CSS网页美化

5.1 DIV元素定位

5.1.1 初识DIV

在第2章中曾介绍过DIV标签，相信读者对它已不陌生。DIV是一个块状标签，是XHTML中指定的专门用于布局设计的容器对象。如果说table是传统表格式布局的核心对象，DIV则就是CSS布局的核心对象，使用CSS布局的页面不再依赖于表格，只需要依赖DIV和CSS，故Web标准布局通常又称为DIV+CSS布局。

在代码中应用＜div＞＜/div＞标签即可插入一个DIV对象，DIV对象中可以容纳段落、标题、图像、音视频、动画等各种HTML元素，DIV的作用只是把内容标识为一个区域，并不负责其他事情，其本质只是一个没有任何特性的容器，为内容添加样式则由CSS来完成。DIV对象在使用的时候，与其他HTML元素一样，可以加入其他属性，但为了实现内容与表现的分离，通常只采用id和class属性两种形式来进行样式编写。在实例5_1.html添加两个DIV标签，并通过id选择器设置CSS规则，使DIV具有不同颜色背景，代码如下：

```
<html>
<head>
<title>插入 Div</title>
<style type = "text/css">
#left {
    background - color: #F90;
}
```

```
#right {
    background-color: #990;
}
</style>
</head>
<body>
<div id="left">内容 1</div>
<div id="right">内容 2</div>
</body>
</html>
```

该代码的浏览效果如图 5-1 所示，从浏览效果来看，在 CSS 规则没有设定宽度的情况下，无论如何调整浏览器窗口，每个 DIV 对象均占据一行，且不允许其他对象与其在同一行中并列显示。实际上，DIV 就是一个块级标签。

图 5-1　插入 Div 标签

在 CSS 布局页面时，HTML 标签根据表现可分为两类：

- 块状标签(block)。块状标签可以理解为容器，如 div、hr、p 等标签，一般为矩形，可以容纳内联标签和其他的块状标签，在默认显示状态下将占据整行，排斥其他标签与其位于同一行。块状标签的 width、height、padding 和 margin 都可以进行有效设置。
- 内联标签(inline)。内联标签又称行内标签，如 span、a、img、iframe 等标签。正好与 block 相反，没有固定形状，可以容纳文本和其他内联标签，它允许下一个对象与其本身在同一行中显示。内联标签的 width 和 height 不起作用，其宽度和高度由自身容纳的文字或图片决定，特殊情况下高度可以通过 line-height 设置。padding 和 margin 在左右方向有效。在宽度允许的情况下，一行可以容纳多个内联标签。

根据 CSS 规范的规定，每个 HTML 标签都有一个默认的 display 属性，用于确定标签的类型，如 DIV 标签默认 display 属性值为 block，span 标签的默认 display 属性值为 inline。根据这个原理，如果要实现内联标签向块级标签的转换从而可以设置 width 和 height 限制，只要将内联标签的 display 属性设置为 block 即可，反之也有效。

DIV 作为一个块级标签，从页面效果看本身与样式没有任何关系，样式需要编写 CSS 来实现。这种与样式无关的特性，使得 DIV 在网页设计中具有巨大的可伸缩性，设计者可以根据自己的想法改变 DIV 的样式，而不再拘泥于传统布局单元格固定模式的束缚，如实例 5_2.html 中，将 id 为 left 的 DIV 设置宽度为 30%，向左 float 浮动，两个 DIV 高度均为 100px，则效果如图 5-2 所示。因此，在 DIV+CSS 布局中，所需要的工作可以简单归纳为两

个步骤：首先利用 DIV 将内容标注出来，然后为 DIV 添加内容并编写所需要的 CSS 样式。

图 5-2　DIV 标签分栏

5.1.2　DIV 布局与嵌套

在布局页面时，为了保证网页不出现水平滚动条，应首先考虑页面内容的布局宽度，并保证页面整体内容在页面居中。目前浏览者的显示分辨率最小为 1024×768 像素，若采用 DIV 元素布局页面宽度时，通常以 width 属性不超过 1002 像素为最佳。

要保证页面整体内容水平居中，方法有多种，常用的方法是用 CSS 设置 DIV 的左右边距，即设置 margin-left 属性和 margin-right 属性值为 auto 时，左右边距将相等，就达到了 DIV 水平居中的效果。

如图 5-3 所示，为了产生一个整体布局，实例 5_3.html 中利用 DIV 产生了一个头部、导航、中部和底部结构，并为每个 DIV 分别定义了 top、nav、mid 和 footer 的 id 名称作为识别。为了方便浏览看到 DIV 的浏览效果，实例中为 DIV 设置了宽度和背景颜色以供区别，4 个 DIV 默认垂直排列并水平居中显示。

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
<title>DIV+CSS 典型布局</title>
<style type="text/css">
body{margin: 10px;
    padding: 0px;}
#top,#nav,#mid,#footer{
    width:1000px;
    margin:0 auto;}
#top,#footer{ height: 35px;
    background-color: #9DD8FF;
    color: #F00;
    font-weight: bold;}
#nav{ height: 35px;
    background-color: #FC6;}
#mid{ height: 100px;
    background-color: #9C3;}
</style>
</head>
<body>
```

```
<div id = "top">顶部</div>
<div id = "nav">导航</div>
<div id = "mid">中部</div>
<div id = "footer">底部</div>
</body>
</html>
```

图 5-3　DIV 垂直排列并水平居中

CSS 代码中 # top、# nav、# mid、# footer 的共同宽度属性 width 统一定义，并通过“margin：0 auto;”对设置边距的 magrin 属性做了进一步优化，属性前面的 0 代表上边距和下边距为 0 像素，auto 代表左边距和右边距为自动，0 和 auto 之间采用空格分隔，这种 CSS 样式定义方法在所有的浏览器中都是有效的。

对于水平居中还可以利用 body 标签属性“text-align:center;”设置所有对象居中，然后再将主体 DIV 容器属性设置“text-align:left;”内容左对齐来实现，CSS 代码定义如下：

```
body {
    text - align: center;
}
# mid {
    width: 1000px;
    text - align: left;
}
```

类似于用表格布局页面，为了实现复杂的布局结构，或者为了内容的需要，多数情况下可能在 mid 中使用左右栏的布局，因此在 mid 中增加了两个 id 分别为 list 和 content 的 DIV，这两个 DIV 是并列关系，而它们都处于 mid 中，从而与 mid 形成了一种嵌套关系，其嵌套布局代码如下：

```
<div id = "top">顶部</div>
<div id = "nav">导航区</div>
```

```
<div id = "mid">
    <div id = "list">列表区</div>
    <div id = "content">内容区</div>
</div>
<div id = "footer">底部</div>
```

实例中 list 和 content 被样式控制为左右显示，设置背景和高度后，最终的页面布局显示效果如图 5-4 所示。网页布局可以由嵌套的 DIV 来构成，无论是多么复杂的布局方法，都可以使用 Div 之间的并列和嵌套来实现。

图 5-4　Div 嵌套布局

5.1.3　浮动定位与清除

第 5.1.1 节提到过块状标签和内联标签的不同，知道作为一个块状标签，DIV 在水平方向上会自动伸展，直到包含它的元素的边界；而在垂直方向上和其他同级元素依次排列，并能并排。但在使用 float 浮动属性后，块级元素的表现就会有所不同。

浮动定位是 CSS 布局中非常重要的手段之一。float 浮动属性可以设置对象左右移动，直到其边缘碰到父元素的边框或另一个浮动元素边缘。CSS 中包括 DIV 在内的所有元素都可以以浮动方式进行显示，其优点是浮动框不在文档的普通流中，这使得内容的排版变得简单，而且具有良好的伸缩性。

float 浮动属性值可以设置为 left、right、none 和 inherit。如果将 float 属性的值设置为 left 或 right，元素就会向其父元素的左侧或右侧边缘浮动。如果设置属性值为 inherit，则表示继承父元素的属性。如果一行有足够的宽度可以容纳两个 DIV 的宽度，则两个 DIV 的内容将会并列于一行显示。这里介绍浮动的几种可能形式，例如页面中两个 DIV 的布局规则如下。

```
#div1 {
    background - color: #CC6;
    height: 300px;
    width: 300px;
    float: left;
}
#div2 {
```

```
        background-color: #FC3;
        height: 400px;
        width: 500px;
    }
```

这里设置了元素 div1 的 float 属性值为 left，第二个元素 div2 的 float 属性为默认值 none，则 div1 脱离文档流并向左移动，直到其边缘碰到包含框（这里是 body 标签）的边缘为止，div2 会顶到原 div1 的位置，其左边框与 div1 的左边框重合，而文字会围绕着 div1 排列。显示效果如图 5-5 所示。

图 5-5　只设置 div1 的 float 为 left 效果

如果设置 div1 元素 float 为 left，而元素 div2 也设置“float:left”，此时 div2 左侧与 div1 的右侧对齐，如图 5-6 所示。

如果设置元素 div2 向右浮动，即设置“float:right”属性，则元素 div2 将紧挨其父元素（这里是 body）右边缘对齐，更改后的效果如图 5-7 所示。

当然，两个 DIV 元素的位置也可以互换，只需要设置 div1 元素的“float:right”和 div2 元素的“float:left”即可，如图 5-8 所示。可见通过设置不同的浮动属性值，可以灵活地定位 DIV 元素的位置，以达到布局网页的目的。

为了更加灵活地定位 DIV 元素，CSS 提供了 clear（清除）属性，用来消除元素的文档流不被影响。clear 属性值有 none、left、right 和 both，默认值为 none。设置“clear:left”可清

图 5-6　两个 div 的“float:left”效果

图 5-7　两个 DIV 左右浮动效果

除 float 对左侧的影响，“clear:right”将清除 float 对其右侧的影响。倘若左右都有浮动的块元素，而新的块元素两侧都不希望受到影响，则可以设置“clear:both”。如实例 5_6.html 所示，div1 和 div2 分别设置 float 为 left 和 right，div3 两端分别受到 div1 和 div2 的 float 影响。若对 div3 添加“clear:left”规则清除左侧浮动影响后，效果如图 5-9 所示。若对 div3 添加“clear:both”规则清除所有浮动影响后，效果如图 5-10 所示。

在 CSS 整体布局中，通常最下端的 footer 部分需要设置 clear 属性，从而消除正文部分排版方法对其的影响。

图 5-8　两个 Div 互换浮动方向

图 5-9　仅清除左侧 float 影响

图 5-10　清除两端 float 影响

5.1.4 position 属性定位

position 从字面意思就是指定块的位置。在 CSS 布局中,position 属性非常重要,很多特殊容器的定位必须用到 position 来完成。position 的属性值包括 absolute(绝对)、relative(相对)、static(静态)、fixed(固定)和 inherit(继承)。其中 static 是默认值,它表示块保持在原本应该在的位置,按照在 HTML 中出现的顺序显示,没有任何移动的效果。fixed 本质上和设置 absolute 一样,只是以浏览器窗口为基准进行定位,块元素不能随着浏览器的滚动条向上或向下移动。inherit 表示从其父元素继承得到。

配合 position 属性使用的,还有 top、right、bottom、left 这 4 个 CSS 属性值,分别表示块元素距离页面或父元素边框的距离(position 取值为 absolute 时),或各个边界离原来位置的距离(position 取值为 relative 时)。这 4 个属性只有当 position 属性设置为 absolute 或则 relative 时才能生效。

这里需要重要介绍的是 absolute(绝对)和 relative(相对)定位。

1. absolute 定位

绝对定位使块元素从 HTML 标准流中分离出来,并把它送到一个完全属于自己的定位中。使用绝对定位的 DIV 元素前面的或者后面的元素会认为 DIV 并不存在,丝毫不影响它们的布局。在计算机显示中,把垂直于显示屏幕的方向称为 Z 方向,CSS 绝对定位的 z-index 属性就对应这个方向,z-index 的值越大,容器就越靠上。简单地说,使用"position: absolute"后,元素就浮在网页上面了,因此,绝对定位常用于将一个元素放到固定位置。例如实例 5_7.html,代码如下:

```
#Div1 {
    height: 200px;
    width: 455px;
    background-color: #CC3;
}
#Div2 {
    height: 200px;
    width: 455px;
    background-color: #FC6;
    position:absolute;
    top:60px;
    left:80px;
}
```

Div2 的 position 属性设置为 absolute 时,就已经不再从属于父块 body,其左边框相对于页面左边框的距离为 80px,上边框距离页面上边框的距离为 60px,这些都是通过 left 和 top 属性设置的。效果如图 5-11 所示。

如果将 Div1 的 position 属性值设置为 absolute,并且调整了它的位置,而 Div2 不设置 position 定位。此时 Div2 便移动到页面最左上端,占据 Div1 原来的位置,而 Div1 则显示在最前方。CSS 代码如下:

图 5-11　Div2 设置 absolute

```
#Div1 {
    height: 200px;
    width: 455px;
    background-color: #CC3;
    position:absolute;
    top:60px;
    left:80px;
}
#Div2 {
    height: 200px;
    width: 455px;
    background-color: #FC6;
}
```

修改后的效果如图 5-12 所示。

图 5-12　仅 Div1 设置 absolute

如果将两个 DIV 的 position 属性同时设置为 absolute,这时两个块元素都按照各自的属性进行定位。两个块元素重叠的部分,Div2 位于 Div1 的上方,这都是因为 CSS 默认后加入的 Div 元素 Z 值大于前者,会覆盖之前的元素。效果如图 5-13 所示。

图 5-13　两个 Div 同时设置 absolute

2. relative 定位

当将块的 position 属性设置为 relative 时,与设置为 absolute 完全不同,这时子块是相对于其父块来作为参照对象偏移定位,而不是相对于浏览器窗口,并且相对定位的块元素脱离标准流浮上来后,无论是否进行移动,其所占的位置仍然留有空位,后面的无定位块元素不会移动上来,因此,移动元素会导致覆盖其他框。相对偏移的方向及幅度由 top、right、bottom、left 属性联合指定,如实例 5_8.html 所示,代码如下:

```
#Div1 {
    height: 200px;
    width: 500px;
    background-color:#CC6;
    position:relative;
    top:40px;
    left:60px;
}
#Div2 {
    height: 200px;
    width: 500px;
    background-color:#FC6;
}
```

Div1 的 position 属性设置为 relative,其位置将以自身起点为基准向上和向右发生偏移,偏移量分别通过“top:40px”和“left:60px”来设置。而 Div2 没有设置任何与定位有关的属性,它还在原来的标准流位置上,并没有像图 5-12 那样移动到页面最左上端并占据原 Div1 的位置。如图 5-14 所示为仅设置了 Div1 的 position 属性为 relative 的效果。

图 5-14　仅 Div1 设置为 relative

如果将 Div1 和 Div2 的 position 属性均设置为 relative，情况又有所不同。这时两个块元素都会相对于它原本的位置，通过偏移指定的距离，到达新的位置，但仍在 HTML 标准流中，其偏移对其他块元素没有任何影响。如图 5-15 所示，这里 Div2 设置为“top:20px”、“left:-30px”，可见偏移量也支持负值。

图 5-15　两个 Div 同时设置 relative

5.2 盒模型理论

5.2.1 盒模型

盒模型(Box Model)是从CSS诞生之时便产生的一个概念,是关系到设计中布局排版和定位的关键问题。任何一个选择符都遵循盒模型规范。

在学习盒模型之前,我们先搞清楚一些概念。假设我们将照片放入一个相框中,对于照片来说,就有了一个“边框”,我们称之为“border”;照片和相框之间通常会有一定的“留白”,我们称之为“padding”;相框彼此之间一般不会紧贴摆放,它们之间相隔的“距离”称之为“margin”。这样的形式存在于生活中的各个地方,如电视机、窗户等,通常这些矩形对象占据的空间都要比单纯的内容要大,就像一个装了东西的“盒子”,称之为“Box”。所谓“盒模型”,就是把每个HTML元素(特别是块级元素)看作一个装了东西的盒子,盒子里面的内容到盒子边框之间的距离即为填充(padding)、盒子本身有边框(border)、而盒子边框外和其他盒子之间还有边距(margin),如图5-16所示。

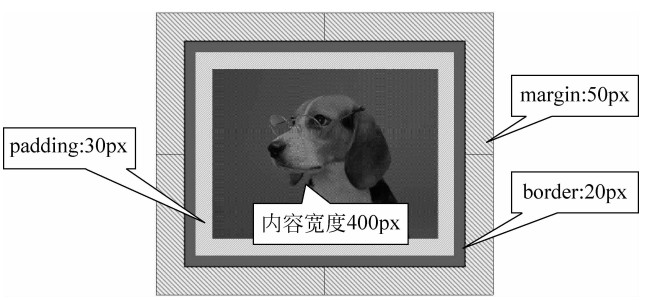

图5-16 盒模型图解

浏览器将每一个HTML标签都作为盒子来处理,但不是所有盒子都是相同的。如p、h1、div等块级标签显示为一块框内容,而span、a等内联标签则显示为一行内框内容,但可以通过display:block属性改变生成框的类型,让内联标签表现得像块级标签一样。

在CSS中,可以通过设定width和height来控制盒子content的大小,并且对于任何一个盒子,都可以分别设置四边各自的border、padding和margin。一个元素的实际宽度=左边距+左边框+左填充+内容宽度+右填充+右边框+右边距。默认情况下,盒子的边框是无,背景透明,所以默认情况下看不到盒子,只有通过CSS样式设置才可以勾勒网页布局。因此,所谓的CSS布局,就是关注这些盒子在页面上如何摆放、如何流动、如何嵌套的问题,只要利用好盒子的这些属性,就能够实现各种各样的排版效果。

5.2.2 边框

边框(border)是指围绕在元素周围的直线,它就像表格一样,可以将文字、图片等网页元素包装起来。边框的CSS样式不但影响到盒子的尺寸,还影响到盒子的外观。每个border可以通过三种不同属性进行控制:border-style(边框样式)、border-width(边框宽度)、border-color(边框颜色)。

border-style 属性主要用来设定 HTML 标签的上下左右边框的风格。具体取值如表 5-1 所示。

表 5-1　border-style 取值说明

样式取值	说　明	样式取值	说　明
none	没有边框	groove	槽线式边框
dotted	点线式边框	ridge	脊线式边框
dashed	破折线式边框	inset	内嵌效果边框
solid	直线式边框	outset	突起效果边框
double	双线式边框		

border-style 是一个复合属性，可以同时取 1～4 个值，4 条边框按照上、右、下、左的顺时针方向调用取值。如果希望为某一个边框设置样式，可以使用单边边框样式属性（包括 border-top-style、border-right-style、border-bottom-style 或 border-left-style）进行设置。如“border-style：solid dotted；”表示上下边框为实线，左右边框为点线。

border-width 用来控制边框的粗细，可以直接使用度量单位（如 2px 或 0.1em）来指定，也可以使用 thin（细边框）、medium（中等边框，默认值）、thick（粗边框）三个关键字之一。border-width 也是一个复合属性，取值方式同 border-style 相同，也可以单独设置四条边框的宽度。

border-color 属性用来设置边框的颜色，同样也可以设置单条边框的颜色，也可以设置四条边框的颜色，颜色的值可以使用十六进制或 RGB 值，也可以使用命名颜色。border-color 也是一个复合属性，取值方式与边框其他属性相同，不再赘述。

边框的三个属性其实可以综合起来，用 border 一条语句代替，如“border：2px solid ＃F00；”一行代码表述了三种属性。如图 5-17 所示，这个样式设置了一条 2px 的红色实心边框，而且这三种属性的编写顺序没有先后，建议读者固定自己熟悉的写法。

图 5-17　border 属性设置

5.2.3 边距

边距(margin)设置元素和元素之间的距离,与 border 属性类似,包括 margin-top、margin-bottom、margin-left、margin-right 及复合边距属性 margin,如果要设置复合边距属性,必须按照上、右、下、左顺时针顺序,不能乱序。

边距的取值有三种方式:长度、百分比和 auto。长度中的像素和 em 比较常用;使用百分比时相当于上级元素宽度的百分比,允许使用负值,如果调整了浏览器的尺寸,这个值也会发生相应变化;auto 就是自动设置边距,这是默认值。

5.2.4 填充

填充(padding)用来控制边框和内容之间的空白距离,类似于 HTML 中表格单元格的填充属性。padding 的所有属性和 margin 属性类似,既可以使用复合属性,也可以使用单边属性,可以接受长度和百分比,但不能使用负值。其百分比值是相对于父级元素 width 计算,如果父级元素 width 改变,padding 宽度也会随之改变。

不同浏览器的 margin 和 padding 的默认值不同,一般情况下最好将所有标签的 margin 和 padding 值初始化为 0,即:

```
body, html {
    margin: 0;
    padding: 0;
    }
```

这里将 margin 和 padding 属性在一个实例中进行体现,代码如下:

```
<html>
<head>
<title>边框边距和填充</title>
<style type = "text/css">
#div1 {
    border:2px solid #F00;
    margin:10px auto;
    padding:20px;
    width:500px;
    height:300px;
    background - color: #CC6;
}
</style>
</head>
<body>
<div id = "div1">边框样式、宽度和颜色设置</div>
</body>
</html>
```

网页代码浏览效果如图 5-18 所示。

图 5-18　margin 和 padding 属性设置

在 CSS 中还存在一个边距折叠的问题，当元素的底部边距碰到另一个元素的顶部边距时，浏览器不是简单地把这两个边距相加，而是取它们中间较大的值。如一个元素 bottom margin 为 20px，另一个相连的元素 top margin 为 10px，则这两个元素的距离就为 20px，而不是 30px，两个边距实际上变成了一个边距。这种问题只发生在垂直边距上，水平边距不会发生这种现象，通过给元素添加 border 或添加 padding 使两个 margin 分隔开可以避免这种情况的发生。

5.3　DIV＋CSS 常见布局

CSS 布局完全有别于传统的布局习惯，它首先将页面在整体上进行＜div＞分块，然后对每个分块进行 CSS 定位，最后再在各个块中添加相应的内容，并对其进行美观设计，这就是 DIV＋CSS 的基本过程。通过 CSS 布局的页面，完全可以通过更新 CSS 属性来重新定位各个版块的位置，如利用 float 和 position 属性可以轻松地移动各个块，从而实现让用户动态选择界面的功能。像页面的左右对调、网页的背景与色调变换等，如果采用传统表格排版，其工作量相当于重新制作一个页面，采用 CSS 布局就相对比较简单。同时，DIV＋CSS 布局方式使得内容与表现完全分离，美工在修改页面时不需要过于关心任何后台操作的问题，可见 CSS 布局具有强大的魅力。

读者可能注意到，互联网上关于 Web 标准的页面大多不是很复杂，这是因为 Web 标准的页面更关注结构和内容，实际上它与网页的美观没有根本冲突。当然，即使是很复杂的网页，也都是一个模块一个模块逐步搭建起来的，本节将从最基本的布局入手，向读者介绍 DIV＋CSS 布局方法。

5.3.1 单行单列布局

单行单列布局是所有布局的基础,也是最简单的布局结构形式。其宽度分固定宽度和自适应宽度两种布局形式,在实际网站创作中应用相当普遍。

1. 固定宽度居中

宽度固定且居中的布局是网络中最常用的布局方式之一。在传统的表格布局方式中,使用表格的 align=center 属性设置可以实现布局居中。使用 CSS 方法也可以实现内容的居中,其解决方法主要有三种思路,下面分别予以实现。

第一种思路,只要设置盒模型的"margin:0 auto;"就行了,即设置上下边距为 0,左右边距自动对齐。如实例 CSS 代码定义:

```
#main {
    width:1000px;
    height:300px;
    margin:0 auto;
}
```

第二种思路,利用 body 或 html 标签属性"text-align:center;"设置所有对象居中,然后再将主体 DIV 容器属性设置"text-align:left;"内容左对齐来实现,CSS 代码定义如下:

```
body {
    text-align: center;
}
#main {
    width: 1000px;
    text-align: left;
}
```

第三种思路,对主体 Div 容器采用 relative 定位方式,设置"left:50%"属性将其左边框移至页面居中处,再利用"magrin-left:-500px;"(假设容器宽度为 1000px)将容器向左拉回宽度一半的距离即可。CSS 代码如下:

```
#main {
    position: relative;
    left:50%;
    margin-left: -500px;
    width: 1000px;
}
```

2. 宽度自适应居中

宽度自适应布局能根据浏览器窗口的大小,自动改变其宽度或高度值,是一种非常灵活的布局形式,对于不同分辨率的显示器能提供最好的显示效果。单列宽度自适应布局只需

要将宽度由固定像素值改为百分比值的形式即可。如“width:75%”,自适应布局的优势是当浏览器窗口大小改变时,DIV 块的宽度将保持浏览器当前宽度的 75%,如实例 5_10.html 中 CSS 代码定义所示,代码如下:

```
<style type="text/css">
#main {
    width:75%;
    height:200px;
    background-color: #CC6;
    margin:auto;      /*左右对齐,自动居中*/
}
</style>
```

当一个盒模型不设置宽度时,它默认是相对于浏览器显示的。宽度自适应更广泛地应用在二列及其他多列布局形式中部分布局块自动调整的排版方式,在后续的章节中将逐步介绍。

5.3.2 二列式布局

1. 二列固定宽度居中

对于宽度固定的二列式布局,一般可利用前面提到的 float 属性或 CSS 的 position 定位属性,结合 margin 属性来实现布局定位,并调整两个布局块之间的距离。这些在前面讲授 float 和 position 属性时有过详细介绍,这里就不再赘述。但要实现居中效果,就要把两个布局块放在一个盒子里,然后让这个盒子居中就可以达到居中效果了,如实例 5_11.html 代码所示,代码如下:

```
<style type="text/css">
#content {
    width:900px;
    height:200px;
    margin:0 auto;
}
#left {
    width:150px;
    height:200px;
    background-color:#FC6;
    float:left;
}
#right {
    width:750px;
    height:200px;
    background-color:#9C6;
    margin-left:150px;     /*此处也可以设置 float:right;效果类似*/
}
</style>
</head>
```

```
<body>
<div id="content">
<div id="left">左列固定宽度</div>
<div id="right">右列固定宽度</div>
</div>
</body>
```

网页代码运行效果如图 5-19 所示。

图 5-19　二列固定宽度居中

2. 左列固定、右列自适应宽度

通过以百分比的方式设置二列宽度，结合 float 属性可以实现二列宽度自适应。在实际应用中，左列宽度固定、右列宽度根据浏览器窗口大小自动适应的布局方式应用最为普遍。其实现也很简单，只需设置左列宽度为固定值，右列不设置任何宽度值，并且右列不浮动即可，如实例 5_12.html 相关 CSS 代码如下：

```
<style type="text/css">
#left {
    float: left;
    width: 384px;
    height: 240px;
    background-color: #FC6;
}
#right {
    height: 240px;
}
</style>
</head>
<body>
<div id="left"><img src="img/picb.jpg" width="384" height="240"/></div>
<div id="right"><p>创意是什么(省略)</p></div>
</body>
```

这里设置左列固定宽度为 384px，而右列根据浏览器窗口大小的变化自动适应调整宽度，在 DIV 中插入相关网页元素后，调整浏览器窗口大小，当右列内容在设置高度“height: 240px”内无法完整显示后，则会显示在左列下方，效果如图 5-20 所示。

图 5-20　左列固定、右列自适应宽度

5.3.3　三列式布局

1. 左右固定、中间宽度自适应

在实际网页应用中，三列式布局一般要求左列宽度固定并居左显示，右列宽度固定并居右显示，中间列能自适应变化。如实例 5-13. html 源代码所示，设置左列 left 的浮动为“float:left”，右列 right 的浮动为“float:right”，中间 center 的 width 值为 auto，其代码如下：

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
<title>左右列固定、中间列自适应</title>
<style type="text/css">
#left {
    height: 300px;
    width: 200px;
    background-color:#CC6;
    float:left;
}
#center {
    height: 300px;
    width:auto;
    background-color:#FC6;
    margin:0 100px;
}
#right {
    height: 300px;
    width: 100px;
    background-color:#CC6;
    float:right;
}
```

```
</style>
</head>
<body>
<div id="left">左列固定宽度 200px</div>
<div id="right">右列固定宽度 100px</div>
<div id="center">中间列自适应宽度</div>      /* 请注意 3 列 Div 的插入顺序 */
</body>
```

网页代码运行效果如图 5-21 所示。

图 5-21　左右固定、中间宽度自适应

左右固定、中间宽度自适应三列式布局也可以采用绝对定位的方法。绝对定位是根据整个网页的浏览窗口来定位的，而前面采用的浮动定位方式是由浏览器根据对象的内容自动进行浮动方向的调整定位的。采用绝对定位的 CSS 代码规则定义如下：

```
#left {
    position:absolute;
    left:0;
    height: 300px;
    width: 200px;
    background-color:#CC6;
}
#center {
    height: 300px;
    width:auto;
    background-color:#FC6;
    margin:0 100px 0 200px;
}
#right {
    position:absolute;
    right:0;
    height: 300px;
    width: 100px;
```

```
    background-color:#CC6;
}
```

在上述代码中，将左列的 DIV 绝对定位，并设置宽为 200px，紧贴页面左侧；右列的 DIV 也是绝对定位，100px 宽度，紧贴右侧边缘，中间的列宽度 auto，且设置上右下左的边界为“margin:0 100px 0 200px;”这样就实现了三列并列放置的效果。但在部分浏览器中三列边缘会因为边界和填充的问题出现空白，为了达到整体最好的效果，这里还要定义 body 和 html 标签的边界和填充都为 0，网页效果同图 5-21 所示，CSS 代码如下：

```
body,html {
    margin:0;
    padding:0;
}
```

2. 三列固定宽度居中

将左右固定、中间宽度自适应三列式布局的实例 5_13.html 做适当修改，把三列放在一个＜div＞标签中，然后让这个＜div＞标签居中，就可以实现三列固定宽度居中的效果，代码如下：

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
<title>三列固定宽度居中</title>
<style type="text/css">
#w {
    width:900px;
    margin:0 auto;
}
#left {
    height: 300px;
    width: 200px;
    background-color:#CC6;
    float:left;
}
#center {
    height: 300px;
    background-color:#FC6;
    margin:0 100px;
}
#right {
    height: 300px;
    width: 100px;
    background-color:#CC6;
    float:right;
}
</style>
```

```
</head>
<body>
<div id = "w">
<div id = "left">左列固定宽度 200px</div>
<div id = "right">右列固定宽度 100px</div>
<div id = "center">中间列自适应宽度</div>
</div>
</body>
```

3. 顶行三列式布局

这里的顶行三列式布局是指顶行自适应宽度,第二行左右列固定宽度,中间列自适应宽度的布局效果,这是网页设计中最常见的网页布局效果。这里一共用到 5 个 div,分别代表顶行、第二行的左列、右列、中间列及包含第二行 3 个 div 的外围列。如实例 5_15 所示,代码如下。

```
<html>
<head>
<meta http-equiv = "Content-Type" content = "text/html; charset = utf-8"/>
<title>三列固定宽度居中</title>
<style type = "text/css">
#top {
    width:1000px;
    margin:0 auto;
    height:50px;
    background-color: #CDC;
}
#w {
    width:1000px;
    margin:0 auto;
}
#left {
    height: 300px;
    width: 200px;
    background-color:#CC6;
    float:left;
}
#center {
    height: 300px;
    background-color:#FC6;
    margin:0 100px;
}
#right {
    height: 300px;
    width: 100px;
    background-color:#CC6;
    float:right;
}
</style>
```

```
</head>
<body>
<div id="top">这是顶行居中显示的列</div>
<div id="w">
<div id="left">左列固定宽度 200px</div>
<div id="right">右列固定宽度 100px</div>
<div id="center">中间列自适应宽度</div>
</div>
</body>
```

网页代码运行效果如图 5-22 所示。读者也可以在网页底部再添加一个 id="footer"的＜div＞并清除上面 div 标签的浮动影响，从而可以完成一个常见完整 Div 布局效果。

图 5-22　顶行三列式布局

5.4　CSS 网页美化

5.4.1　背景样式控制

网页的背景属性是网站设计的一个重要步骤，通过设置背景颜色或背景图像，能给网页带来丰富的视觉效果，有时候甚至网页背景图的风格决定了网站的整体风格。CSS 对元素的背景设置提供了很多的属性定义，包括背景颜色、背景图像、重复方式及其定位参数等。本节将通过实例，分别讲解有关背景样式控制的各个属性。

1. 背景颜色 background-color

背景颜色用来设置对象的背景颜色，其 CSS 语法为：

```
background-color:transparent|color
```

其中，参数 transparent 表示背景色透明，color 指定颜色，默认值是 transparent。颜色值的设置方式和文本颜色值的设置方式一样，可以采用十六进制、RGB 分量或颜色的英文名称等。在 Dreamweaver 中使用“CSS 规则定义”对话框中的“背景”类别中的“背景颜色”选项可以定义 CSS 样式的背景颜色。可以通过为 body 标签定义 CSS 背景规则来定义整个页面的背景，也可以定义 Web 页面中一个具体的 HTML 元素背景。其主要作用在于突出

页面主题、前景色及文字相匹配，如实例 5_16.html 所示，通过设置 body 标签“background-color：＃968F5B;”即背景颜色为＃968F5B，实现背景与图像完美结合，其效果如图 5-23 所示。

图 5-23　网页背景颜色

2. 背景图像 background-image

网页背景除了可以使用各种颜色，同样也可以使用各种图片。背景图在网页设计发展初期只发挥了强调质感和修饰页面的功能，而通过 CSS 可以对背景图像进行精确的控制，包括位置和重复方式等，从而将其功能发挥到了极致。在 HTML 中设计网页的背景图片和设置表格的背景图片都是使用 background 属性，在 CSS 中使用 background-image 属性来设置背景图片，其 CSS 语法为：

```
background - image:none|url(url)
```

其中，参数 url 是指背景图片的地址路径，存放在括号中，既可以使用绝对路径，也可以使用相对路径。对路径使用不是很熟悉的初学者，也可以在 Dreamweaver 中使用“CSS 规则定义”对话框“背景”类别中的“背景图像”选项来定义 CSS 样式背景。

3. 背景图像重复方式 background-repeat

在默认情况下，背景图以平铺的方式出现，同时覆盖整个页面。这种方式并不适合大多数页面。在 CSS 中可以通过 background-repeat 属性设置图片的重复方式，包括 repeat 默认平铺背景、repeat-x 水平重复、repeat-y 竖直重复、no-repeat 不重复等。将 background-image 属性和 background-repeat 属性结合使用，利用背景图的横向或纵向平铺，往往可以得到意想不到的效果。如图 5-24 所示即为利用横向平铺背景图完成的页面效果图。

4. 背景图像定位 background-position

在传统的表格布局中没有办法提供精确到像素级别的背景定位方式，所设置的背景图片都是从设置了 background 属性的标记(例如 body 标记)的左上角开始出现的。在实际制

图 5-24　背景图片横向平铺

作中，CSS 可以通过 background-position 属性调整背景图片的位置。例如，如果希望背景图片出现在页面的右下角时，可以将 background-position 的值设定为 bottom right 来实现。如图 5-25 所示。

图 5-25　背景定位

除了 bottom right 外，background-position 的值还可以设置多种类型，CSS 中可以通过 3 种不同的方法来设置图片在水平方向和垂直方向的起点，分别是关键字、精确值和百分比。

关键字有两组选项，一组是用来控制水平方向的 3 种定位：left(左)、center(居中)、right(右)；另一组控制垂直方向的 3 种定位：top(顶端)、center(居中)、bottom(底端)。水平方向和垂直方向的关键字可以任意搭配，共产生 9 种不同定位方式。如要把背景图片放

在正中央，可以创建样式“background-position:center center;”即可。

第二种设置方法是精确值，可以使用像素值来定位背景图片。这里要使用两个值，一个用来控制水平方向的位置，另一个用来控制垂直方向的位置。如“background-position:20px 20px;”背景图片左侧和顶端分别距离网页20px。采用这种方法一般不会设定距离网页底端和右边的距离，但可以利用负数值将图片移出部分右边或顶端，从而实现隐藏部分图片效果。

第三种设置方法是使用百分比，如通过代码“background-position:100% 50%;”的设置，可以使背景图像的中心点在水平方向上居右，在竖直方向上则居中的位置，等效于采用关键字 right center 效果。当浏览器的宽度改变时，图片的位置也会发生变化。背景图像的定位功能可以用在图像和文字的混合排版上，将背景图像定位在合适的位置，可以获得最佳的图文效果，如实例5_18.html中，设置“background-position:60% 10%;”的效果如图5-26所示。

图5-26　百分比精确定位背景图像

5. 背景附件 background-attachment

当网页中显示的内容比较多，在浏览器的右侧就会出现滚动条，拖动滚动条向下移动，可以查看更多的网页内容，但网页背景也会随之向上移动。利用CSS的固定背景属性，还可以建立不滚动的背景图像，页面滚动时，背景图像可以保持不变，其语法如下：

```
background-attachment:scroll|fixed
```

其参数分别表示背景图像随对象内容滚动(scroll)或固定(fixed)。scroll是浏览器的默认值，表示背景图片会随着滚动条移动而移动，fixed表示把图片固定在网页背景中的某个位置，不随着滚动条的移动而移动，如实例5_19.html所示，代码中为了出现滚动条，人为增加了网页高度height，当滚动条向下拖动时，可以看到网页背景保持位置固定不变，效果如图5-27所示。

图 5-27　固定背景图片效果

5.4.2　段落及字体样式设计

文字是信息的主要载体方式，是网页最重要的信息表达工具。网页文字阅读的舒适程度直接关系到浏览者是否愿意继续浏览网站。网页中的文字样式涉及文字设计、段落样式及 CSS 滤镜图案文字等，在设计时都需要谨慎考虑。

1. 字体样式设计

在 HTML 中，关于字体样式设计使用＜font＞标签的相关属性，在 CSS 中该标签已不再推荐使用，可以使用 font-family、font-size、color、font-style、text-decoration 等属性进行文本修饰。定义 CSS 文字样式可通过 Dreamweaver 进行可视化操作，在“CSS 规则定义”对话框的“分类”列表中选择“类型”选项，可以完成文本样式的定义，包括字体、大小、颜色、粗细、斜体、下划线、上划线、删除线及英文文本控制等，如图 5-28 所示，具体参数本书在第 4 章做过详细介绍，这里不再详述。

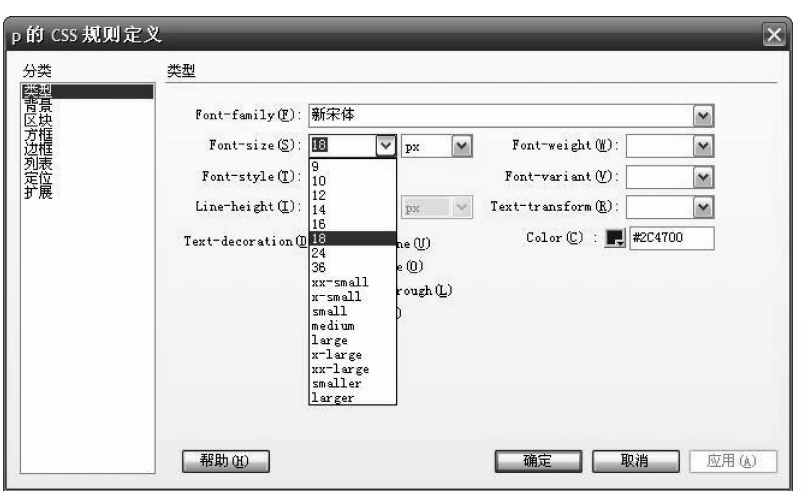

图 5-28　可视化设置文本样式

实例 5_20.html 设置后的 CSS 规则代码如下：

```
#text {
    font-family: "楷体_GB2312";
    font-size: 24px;
    font-weight: lighter;
    text-transform: none;
    color: #069;
    font-style: normal;
    background-image: url(img/bg5.jpg);
}
```

2. 段落样式设计

文字排版必然会涉及段落，CSS 在段落控制方面提供了丰富的样式属性，可用于设置行高、间距、对齐、缩进、文字竖排等，还可以通过定义 CSS 伪对象来实现首字下沉效果。其中行高、对齐、缩进等可以通过打开“CSS 规则定义”对话框“分类”中的“区块”选项来进行设置，这里不再详述。

对于首字下沉的效果，需要定义一个 CSS 伪对象，其 CSS 代码如下：

```
#text:first-letter {
    font-family: "黑体";
    font-size: 2em;
    color: #C00;
    float: left;
}
```

表示针对 #text 中文本的第一个字符进行样式控制，重新设置了文字尺寸为 2em，表示它是其他字符 2 倍大小。使用“float:left”使右边的文本显示在第一个字符的右边，而不是换行。效果如图 5-29 所示。

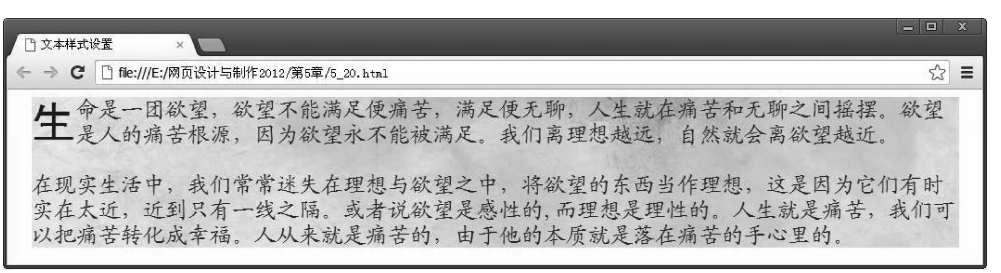

图 5-29　首字下沉效果

文字竖排属性为 writing-mode，其参数设置包括 lr-tb(左右，上下)和 tb-rl(上下，右左)两种。只要浏览器版本支持该属性，一般都能显示正常的效果。文字排版规则格式较多，其他关于文字排版的内容读者可以参考相关 CSS 手册。

5.4.3 图片样式控制

在网页中利用各种各样的图片，能够让人更直观地感受网页要传达给浏览者的信息，也能够充分展现网页的主题并增强网页的美感。CSS 提供了强大的图像样式控制能力，包括图像的边框、对齐方式、图文混排等。本节将通过实例介绍 CSS 的图像样式控制能力。

1. 图像边框设计

在 HTML 中，可以直接通过＜img＞标签的 border 属性值为图片添加边框，从而控制边框的显示和粗细。在 Dreamweaver 的“CSS 规则定义”对话框中，可以通过“边框”分类中的“样式”下拉列表选择边框的样式外观，并可以设置边框线的宽度和颜色。如图 5-30 所示即为通过更改边框样式、宽度和颜色得到的 4 种不同边框效果图。

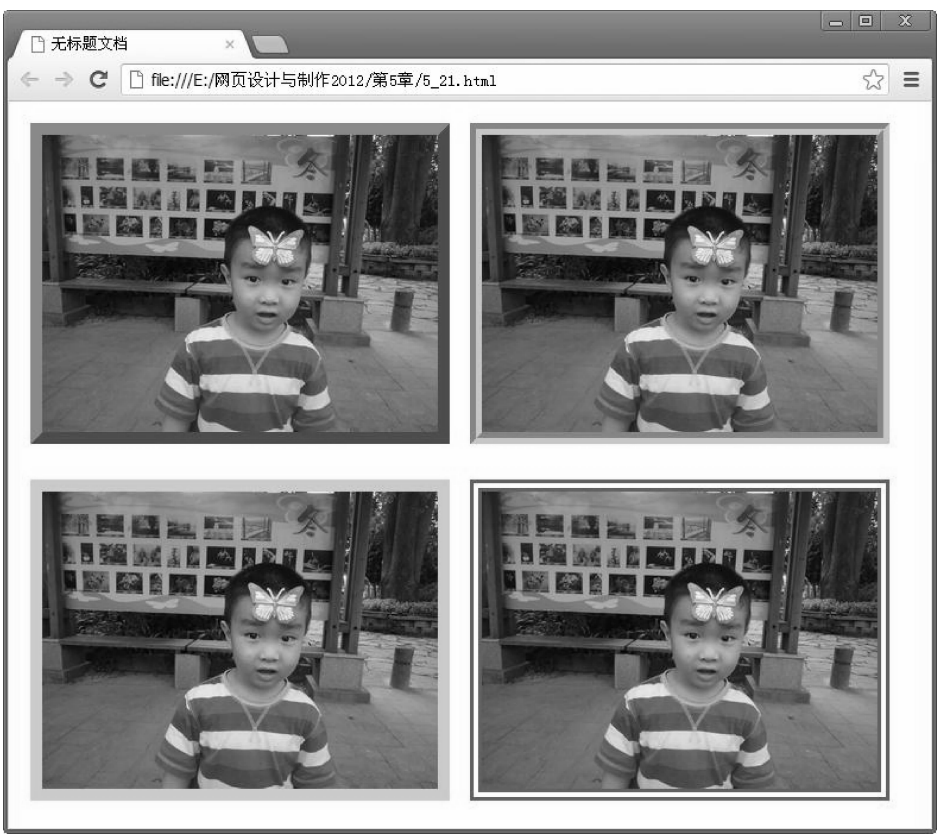

图 5-30　不同边框效果图

4 种不同边框效果对应的 CSS 规则核心代码如下：

```
#div1 img {
    border: 10px outset #B80;
}
#div2 img {
    border: 10px groove #FC0;
}
```

```
#div3 img {
    border: 10px solid #FC3;
}
#div4 img {
    border: 10px double #36C;
}
```

2. 图像大小控制

CSS 也可以像 HTML 一样，通过 width 和 height 两个属性来控制图片的大小，但 CSS 中可以使用更多的值。例如实例 5_22.html 所示，设置 width 为 75%时，图片的宽度将调整为父元素宽度的 3/4，网页源代码如下：

```
<style type="text/css">
<!--
img {
    width: 75%;
}
-->
</style>
</head>
<body>
<img src="img/picc.jpg" alt="缩放"/>
</body>
```

显示效果如图 5-31 所示，当拖动浏览器改变窗口大小时，图片的大小也会相应的发生变化。

图 5-31　图片大小缩放

这里要注意的是，如果只是设置了图片的 width 属性，而没有设置 height 属性时，图片本身会自动等长宽比缩放，如果只是设置了图片的 height 属性效果也是一样。只有同时设置了 width 和 height 属性时才不会等比例缩放。如果设置“width：75%”，设置“height：250px”，当浏览器窗口变化时，高度不会随着图片宽度变化而改变，会出现图片的扭曲现象。

3. 图文混排设计

在网页中图片和文字往往需要同时使用，如果直接在网页文档中插入图像，图像会单独占用一行。这就涉及图文混排的问题，由于段落的 p 元素是块状元素，在 CSS 中通过给图像设置 float 浮动属性可以实现图文混排效果，同时还可以通过 padding 属性设置图片和文字的间距等。如实例 5-23.html 所示，设置图像宽度为 50%，居左，padding 值为 10px，其效果如图 5-32 所示。

图 5-32　图文混排效果

5.4.4　CSS 滤镜

CSS 滤镜并不是浏览器的插件，也不符合 CSS 标准，而是微软公司为了增强浏览器的功能而特意开发并整合在 IE 浏览器中的一类功能的集合，只有 IE 浏览器才有效果，目前在 Google Chrome 和 Mozilla Firefox 中无效果。IE 从 4.0 开始，就提供了一些内置的多媒体滤镜特效，利用它可以为网页增加非常美妙的视觉效果。

CSS 滤镜属性只能用在 HTML 空间元素上。所谓 HTML 空间元素就是他们在网页上定义了一个矩形空间，浏览器窗口可以显示这些空间。滤镜并不是对元素进行处理，而是在浏览器中对使用了该属性的对象进行了一定的修饰，实际上是样式表一个新的扩展部分。

使用这种技术简单的语法就可以把可视化的滤镜和转换效果添加到一个标准 HTML 元素上。

CSS 滤镜属性的选择符是 filter，只要进行滤镜操作，都必须先定义 filter，其他和普通 CSS 语句一样，都十分简单。常见的滤镜属性如表 5-2 所示。

表 5-2　常见的滤镜属性

属性名称	属性解释	属性名称	属性解释
Alpha	设置透明层次	Gray	把图片灰度化
Blendtrans	淡入淡出效果	Invert	反色
Blur	模糊效果	Light	创建光源在对象上
Chroma	专用颜色透明	Mask	创建透明掩膜在对象上
DropShadow	创建对象的固定影子	Revealtrans	随机产生 23 种动态效果
FilpH	水平镜像图片	Shadow	创建偏移固定影子
FilpV	垂直镜像图片	Wave	波纹效果
Glow	加光辉在附近对象的边沿外	Xray	X 光照射效果

【注意事项】

滤镜(filter)只在 IE 浏览器中有效。

1. Alpha 滤镜

Alpha 滤镜用来设置透明度，它把一个目标元素与背景混合，设计者可以指定数值来控制混合的程度。通过指定坐标，还可以指定点、线、面的透明度。其语法结构如下：

```
filter:Alpha(opacity = opacity,finishopacity = finishopacity,style = style,startX = startX,
startY = startY,finishX = finishX,finishY = finishY)
```

其中，opacity 代表透明度等级，取值从 0～100，0 代表完全透明，100 代表完全不透明。style 指定透明区域的形状特征，其中 0 代表统一形状，1 代表线性，2 代表放射状，3 代表长方形。finishopacity 是一个可选项，用来设置结束时的透明度，从而达到渐变效果，取值也是 0～100。startX 和 startY 及 finishX 和 finishY 分别代表渐变透明效果的开始和结束坐标。

通过“CSS 规则定义”对话框中的“扩展”分类上的“滤镜”下拉列表框中选择 Alpha 滤镜，输入参数值设置滤镜样式。效果如图 5-33 所示。

对应的滤镜样式代码为：

```
filter: Alpha(opacity = 0, finishopacity = 100, style = 1);
```

2. Flip 翻转滤镜

Flip 是 CSS 滤镜的翻转属性，它有两种应用，其中 FlipH 代表水平翻转，FlipV 代表垂直翻转。其语法结构很简单，直接使用“filter:FlipH”或“filter:FlipV”，没有任何参数设置。如实例 5_25.html 所示，样式创建成功后，效果如图 5-34 所示。

3. Gray 滤镜

Gray 滤镜能够将一张彩色图片变成黑白图片，给人一种怀旧和回味的感觉。其语法结构非常简单，语法如下：

图 5-33　Alpha 滤镜效果

图 5-34　FlipH 水平翻转滤镜效果

```
filter:Gray
```

直接将该滤镜作用于图片上即可实现相应的效果。

4. Invert 滤镜

Invert 滤镜用于把对象的可视化属性全部翻转,包括色彩、饱和度和亮度值,即反色,相当于照片底片的效果,其语法结构如下:

```
filter:Invert
```

和 Gray 滤镜一样,直接将该滤镜作用于图片上即可实现相应的效果。

5. Xray 滤镜

Xray 射线滤镜和它的名字一样,就是给照片添加 X 光照射的效果,其语法结构如下:

```
filter:Xray
```

很多时候浏览者往往把 Xray 效果和灰度 Gray 效果混淆,如图 5-35 所示,从左到右依次分别添加了 Gray 滤镜、Invert 滤镜和 Xray 滤镜效果,对比可知三者有明显的区别。

图 5-35　Gray 滤镜、Invert 滤镜和 Xray 滤镜效果

6. Chroma 滤镜

Chroma 滤镜用于设置一个对象中指定的颜色为透明色,其语法结构如下:

```
filter: Chroma(Color = color);
```

如实例 5_27.html 网页,在 div 对象中输入文本“网页设计”,设置其颜色为 #FF0000,在“CSS 规则定义”对话框中设置滤镜 Chroma,并输入参数值为 #FF0000,效果如图 5-36 所示,生成的 CSS 规则代码如下:

```
filter: Chroma(Color = #FF0000);
```

图 5-36　Chroma 滤镜效果

7. Wave 滤镜

Wave 滤镜可以为对象添加竖直方向上的波浪效果，也可以用来把对象按照竖直的波纹样式打乱，其语法结构如下：

```
filter: Wave(add = add, freq = freq, lightstrength = lightstrength, phase = phase, strength = strength);
```

add 参数有两个值，用来设置是否显示原对象，取 0 值表示不显示，取非 0 值表示要显示原对象。freq 参数指生成波纹的频率，也就是指定在对象上共需产生多少个完整的波纹。lightstrength 参数是为了使生成的波纹增强光的效果，值可以为 0～100。phase 参数用来设置波浪的起始相角，从 0～100 的百分数值。如该值为 25，代表正弦波从 90°(360×25%)的方向开始。strength 为振幅的大小，取值为自然数。如设置 CSS 规则代码如下：

```
filter: Wave(add = 0, freq = 5, lightstrength = 8, phase = 2, strength = 8);
```

效果如图 5-37 所示。

图 5-37　Wave 滤镜波纹效果

5.5　导航菜单及超链接

在传统的表格布局设计中，通常都是利用表格、文本加超链接来完成导航的设计。在 CSS 布局中，可以使用 ul 列表来制作导航。实际上，导航可以理解为导航列表，导航中的每一个栏目都是一个列表项。因此，本节先围绕项目列表的基本 CSS 属性进行相关介绍，再深入介绍导航和超链接的设置。

5.5.1 列表设计

通常的项目列表主要采用<ul>或者<ol>标记,同时配合<li>标记罗列各个项目。CSS列表属性允许设置列表项标记,可以改变标记的样式、图像及位置等,列表属性包括list-style-type(列表样式)、list-style-image(标记图像)、list-style-position(标记位置)等。

list-style-image属性语法格式为:

```
list-style-image:none|url;
```

其中none不设定图片,url指定图片路径。

list-style-position属性语法格式为:

```
list-style-position:outside|inside;
```

其中默认outside列表标记放置在文本以外,且环绕文本根据标记对齐,inside列表标记放置在文本以内。

list-style-type列表样式属性值比较多,在Dreamweaver的"CSS规则定义"对话框中"列表"分类的"类型"中提供了9种默认样式供选择。在这9种样式中,针对ul无序列表的有4种,分别为disc(圆点)、circle(圆圈)、square(方块)、none(无)。针对ol有序列表,除了none之外,还有5种类型的样式,包括decimal(数字)、lower-roman(小写罗马数字)、upper-roman(大写罗马数字)、lower-alpha(小写字母)、upper-alpha(大写字母)。除这9种样式外,列表样式还包括大小写拉丁字母、日文平/片假名字符或序号以及其他符号等,读者可以查阅相关资料进行选择。

如果希望设置项目符号采用图片的方式,可以使用"list-style-type:none"取消默认的圆点项目符号,接着对li标签设置一个不重复的背景图像,并利用padding-left设置图标与文字的间隔距离。为了防止li标签中的文字压住背景图标,将li标签的左填充属性设置为30px,使背景图像显示处理,实例5_29.html代码如下:

```
<style type="text/css">
#divnav {
    font-family: "新宋体";
    font-size: 24px;
    background-image: url(img/bg6.jpg);
    background-repeat: no-repeat;
    height: 600px;
    width: 960px;
    margin:0 auto;
    padding:20px;
}
ul {
    list-style-type: none;
}
li {
    background-image: url(img/treey.gif);
    background-repeat: no-repeat;
```

```
    background-position: 0px 0px;
    padding-left: 30px;
}
</style>
</head>
<body>
<div id="divnav">
  <ul>
    <li>网页设计与制作</li>
    <li>网页布局与网站构成</li>
    <li>网页设计中的文字设计</li>
    <li>网页设计中的特效设计</li>
    <li>网页色彩设计</li>
  </ul>
</div>
</body>
```

效果如图 5-38 所示。

图 5-38　列表背景图像效果

5.5.2　导航菜单

导航从形式上一般可分为横排导航、竖排导航、下拉导航、图片导航、Flash 导航等，对于门户型网站及多数栏目复杂的网站均以横排导航为主，竖排导航也是网站应用中非常普遍的一种形式。有一些网站的导航形式不是单一的，既包括横排导航，侧面也有竖排导航。总之，导航的设计要根据网站栏目的分类和内容的多少来合理采用导航形式。导航设计的主要原则是引人注目，使用方便。

导航都包含有超链接，所以一个完整的网站导航必须创建超链接样式。本节先主要针对各种类型的导航形式进行讲授，后续章节中再单独针对超链接样式的设计进行介绍。

1. 横排导航

用列表实现导航的 XHTML 源代码如下：

```
<div id = "divnav">
  <ul>
    <li>首页</li>
    <li>博客</li>
    <li>相册</li>
    <li>好友</li>
    <li>留言</li>
    <li>联系</li>
  </ul>
</div>
```

其中，对象 #divnav 可以看作列表的容器。此时，ul 默认为从上到下排列，要实现横排导航，必须定义 li 对象的浮动方式 float 为 left，才能实现同行显示多个列表项。添加对 ul 和 li 对象的背景、填充、边界、宽度和文本对齐方式的设置，同时对 #divnav 定义背景图像，进一步美化导航栏，从而形成如图 5-39 所示的横排导航效果，CSS 规则定义代码为：

```
<style type = "text/css">
#divnav {
    font-family: "黑体";
    font-size: 24px;
    width: 1000px;
    height: 667px;
    margin:0 auto;
    background-image: url(img/bg8.jpg);
    background-repeat: no-repeat;
}
#divnav ul {
    list-style-type: none;
    position: relative;
    left: 50px;
    top: 50px;
    width:500px;
}
#divnav li {
    float:left;
    padding:5px;
    text-align:center;
    background-image: url(img/treey.gif);
    background-repeat: no-repeat;
    padding-left: 30px;
}
</style>
```

图 5-39　横排导航效果

2. 竖排导航

建立相关 XHTML 的列表结构后，将导航项目列表添加到＜ul＞标签中，同时设置页面的背景图像和字体。设置＃divnav 块中＜ul＞的属性，将项目符号设置为不显示，并固定它的位置，接着给＜li＞标记设置一个不重复的背景图像作为项目符号，设置文字和图标的间隔距离 padding-left 的值，但并没有设置 float 浮动属性，则可以实现一行只显示一个块元素标记。相关 CSS 代码如下：

```
<style type="text/css">
#divnav {
    width: 1000px;
    height: 667px;
    font-family: "黑体";
    font-size: 24px;
    background-image: url(img/bg8.jpg);
    background-repeat: no-repeat;
    color: #132A10;
    margin:0 auto;
}
#divnav ul {
    list-style-type: none;
    width:200px;
    position: relative;
    left: 100px;
    top: 50px;
}
#divnav li {
```

```
        border-bottom:1px solid #132A10;
        background-image: url(img/treey.gif);
        background-repeat: no-repeat;
        background-position: 0px 0px;
        padding-left: 30px;
    }
    #divnav li a{
        display:block;
        padding:5px;
        text-decoration:none;
    }
    </style>
```

这里要特别说明的是#divnav li a中的“display:block;”语句，通过该语句，超链接被设置成块元素，当鼠标进入该块的任何部分时都可被激活，而不是仅仅在文字上方时才被激活。此时页面的显示效果如图5-40所示。

图5-40　竖排导航效果

3. Tab标签式导航

Tab标签式导航是横排导航的一种，类似于文件夹标签的样式。这种形式可以让浏览器很清楚地知道目前浏览的是哪一个栏目，因为当前栏目的标签与其他标签会有明显不同的背景或颜色等。

观察标签式导航的特点，可以发现，要使一个栏目成为当前栏目，必须对这个栏目的样式进行单独的设计。假设当前栏目为第三个“相册”标签中，为该li标签加上了“id=select”，即<li id=select>相册。为这个li标签设置CSS样式，代码如下：

```
#divnav li#select {
    background-color:#996;
}
```

对其他导航栏目进行美化，添加一个#content内容块，修改并添加边框属性，添加一个#main块，用于控制页面居中效果。这里并没有针对超链接设置相应规则，在下一小节中会举例介绍，本例完整的CSS代码如下：

```
<style type="text/css">
#main {
    font-family: "黑体";
    font-size: 20px;
    background-image: url(img/bg8.jpg);
    background-repeat: no-repeat;
    margin:0 auto;
    width:1000px;
    height:650px;
}
#divnav {
    width: 500px;
    left: 70px;
    top: 50px;
    position: relative;
    border-right: 2px solid #006600;
    border-top: 2px solid #006600;
    border-left: 2px solid #006600;
    height: 35px;
}
#divnav ul {
    margin: 0px;
    padding: 0 5px;
    display:block;
}
#divnav li {
    list-style-type: none;
    padding-left: 20px;
    padding-right:20px;
    float:left;
    height:30px;
    padding-top:5px;
}
#divnav li#select {
    background-color:#996;
}
#content{
    border-right: 2px solid #006600;
    border-bottom: 2px solid #006600;
    border-left: 2px solid #006600;
```

```
        position: relative;
        top: 50px;
        left:70px;
        width: 480px;
        height: 300px;
        padding: 10px;
        background-color:#996;
    }
    </style>
    </head>
```

在内容块中插入图片,网页最终效果如图 5-41 所示。

图 5-41　Tab 标签式导航

5.5.3　超级链接

超链接是整个网站的灵魂,网页通过超链接实现页面的跳转、功能的激活等,因此,超链接的效果直接影响到用户的浏览感受。在 HTML 语言中,超链接是通过标记＜a＞来实现的,链接的具体地址则是利用＜a＞标记的 href 属性来指定。

在默认的浏览器浏览方式下,超链接统一为蓝色且有下划线,被点击过的超链接则为紫色并且也有下划线。通过 CSS 可以设置超链接的各种属性,包括字体、颜色、背景灯,还可以通过伪类来制作很多动态效果,最简单的莫过于去掉超链接的下划线,代码如下所示。

```
a { text-decoration:none; }
```

其效果无论是超链接本身，还是被单击过的超链接，下划线都被去掉了。除了颜色以外，和其他普通文本没有差别。

利用 CSS 伪类可以制作动态超链接效果，具体属性如表 5-3 所示。

表 5-3　CSS 伪类属性

属　　性	说　　明
a:link	正常浏览状态样式
a:visited	被单击过的超链接样式
a:hover	鼠标经过超链接上时的样式
a:active	在超链接上单击时的样式

CSS 就是利用以上 4 个伪类属性，配合各种属性风格制作出丰富多彩的动态超链接。如实例 5_33.html 所示，在 CSS 控制超链接的代码中，包括对超链接本身、被访问过的超链接以及鼠标指针经过时的超链接进行了样式的修饰，相关 CSS 代码如下：

```
a:link {
    color: #FFFF66;
}
a:visited {
    color: #99FF00;
}
a:hover {
    color: #CCFFCC;
    background-image: url(img/fudiao2.jpg);
    background-repeat: no-repeat;
}
</style>
</head>

<body>
<div id="divtop"></div>
<div id="div1">
  <table height="24">
    <tr>
      <td><a href="box.html">首页</a></td>
      <td><a href="#">博客</a></td>
      <td><a href="#">相册</a></td>
      <td><a href="#">创意</a></td>
      <td><a href="#">图片</a></td>
      <td><a href="#">留言</a></td>
      <td><a href="#">下载</a></td>
      <td><a href="#">联系</a></td>
    </tr>
  </table>
</div>
```

对于激活状态“a:active”,因为当用户单击一个超链接之后,焦点很容易就会从这个链接上转移到其他地方,此时该超链接就不再是“当前激活”状态了。故一般被显示的情况非常少,此例中做了忽略处理。

如果将背景图加入到超链接的伪类设置中,还可以制作出更多绚丽的效果。这里以浮雕效果为例,来看看如何制作比较酷的超链接样式。

首先要利用绘图软件制作浮雕的背景图,作为超链接所在行的背景,并设置为水平重复。同样制作两个宽度固定的浮雕图像,分别在最左边绘制 4px 宽的竖直白线和竖直绿线,作为超链接的正常状态和鼠标经过超链接状态时的背景。完整 CSS 规则代码如下:

```
<style type="text/css">
#divtop {
    background-image: url(img/95top.jpg);
    background-repeat: no-repeat;
    margin:0 auto;
    background-position: center;
    height: 145px;
    width: 960px;
}
#div1 {
    height: 30px;
    width: 960px;
    margin:0 auto;
}
a {
    background-image: url(img/fudiao.jpg);
    background-repeat: no-repeat;
    text-align: center;
    padding-left: 10px;
    display:block;
    text-decoration: none;
}
#div1 table {
    text-align: center;
    width: 100%;
    font-family: "新宋体";
    font-size: 18px;
    color: #FF9;
    font-weight: bold;
    background-image: url(img/fudiaobg.jpg);
    background-repeat: repeat-x;
}
a:link {
    color: #FF6;
}
a:visited {
```

```
    color: #9F0;
}
a:hover {
    color: #CFC;
    background-image: url(img/fudiao2.jpg);
    background-repeat: no-repeat;
}
</style>
```

变化后页面效果如图 5-42 所示。

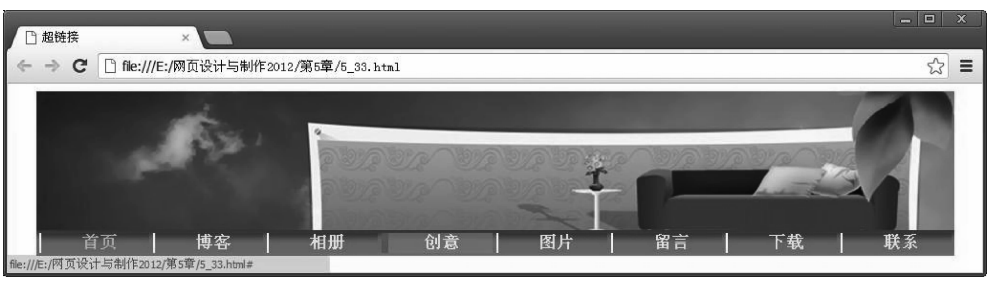

图 5-42　动态超链接效果

5.6　上 机 实 践

5.6.1　利用 float 浮动定位

已知一个网页的 HTML 代码如下：

```
<body>
  <div id="list">
        <ul>
            <li><img src="img/3698.jpg"><br/>优雅绅士</li>
            <li><img src="img/3699.jpg"><br/>粉色迷情</li>
            <li><img src="img/3723.jpg"><br/>童真童趣</li>
            <li><img src="img/3742.jpg"><br/>亲亲宝贝</li>
            <li><img src="img/3723.jpg"><br/>童真童趣</li>
            <li><img src="img/3742.jpg"><br/>亲亲宝贝</li>
            <li><img src="img/3698.jpg"><br/>优雅绅士</li>
            <li><img src="img/3699.jpg"><br/>粉色迷情</li>
        </ul>
</div>
</body>
```

试根据如图 5-43 所示的网页效果，利用 float 浮动定位完成该页面的元素定位，并编写 CSS 规则定义代码。

图 5-43　float 浮动定位

5.6.2　利用 position 属性定位

position 属性可以设置绝对定位和相对定位，分别通过设置 left、top、right、bottom 来完全定位或进行位置偏移，试分别通过 position 的 absolute 和 relative 定位来实现如图 5-44 所示顶部页面效果(提示：可以利用 DIV 的嵌套)。

图 5-44　position 属性定位

5.6.3　编写典型的网页布局

结合前面所学的布局知识，利用 DIV＋CSS 制作如图 5-45 所示的网页布局效果，其布局要求如下：

页面要求有上下 4 行区域，分别用作顶部广告区、导航区、主体区和版权信息区。而主体区又分为左右两个大区，左区域用于文字列表，右区域用于 6 个主体内容区。其中，用 ＃top 代表顶部广告区，＃nav 代表导航区，＃mid 代表主体区，＃left 代表＃mid 所包含的

图 5-45　网页布局结构图

左文字列表区域，#right 代表 #mid 所包含的右边主体内容区，6 个主体内容区用.content 类来表示，#footer 代表版权信息区。

5.6.4　网页元素美化及导航

利用 5.6.3 节编写典型的网页布局，插入网页元素文本及背景图像，插入图像元素，并实践 CSS 滤镜效果，创建导航超链接并利用 CSS 对导航区进行美化处理，围绕一个主题制作一个精美的页面效果。

5.6.5　改造基于 fable 的网页

利用给定的基于 fable 完成的网页及素材，根据实验步骤及所学知识将其转化成基于 Div＋CSS 标准布局的网页效果。